尤度を用いた進化戦略による LLM の最適化

福田創　河原大輔
早稲田大学
so.fukuda@akane.waseda.jp　dkw@waseda.jp

概要

大規模言語モデル（LLM）のアライメントにおいて，進化戦略（ES）は勾配ベースの手法の有力な代替手段として注目されている．しかし，従来の ES はパラメータにノイズを加えた多数のモデルそれぞれでテキスト生成を行う必要があり，計算コストが高いという課題がある．本論文は，生成プロセスを現在のモデルのみに限定し，ノイズ付加モデルでは生成されたサンプルの尤度のみを計算することで最適化を行う Likelihood-Based Evolution Strategies（LBES）を提案する．実験の結果，LBES は従来の ES と比較して，一定の精度に到達するまでの時間を大幅に短縮できることを確認した．一方で，到達可能な最高精度に関しては，特に難易度の高いタスクにおいて従来の ES を下回る傾向も見られた．実装コードは https://github.com/nlp-waseda/lbes にて公開している．

1　はじめに

進化戦略（ES）は，勾配情報を直接必要としないブラックボックス最適化手法であり，近年，大規模言語モデル（LLM）のファインチューニングやアライメントにおいて研究が進んでいる [1, 2, 3, 4]．LLM に適用した ES の一般的な手順を Algorithm 1 に示す．具体的には，パラメータにノイズを加えた多数のモデルを作成し，それぞれのモデルで実際にタスクを実行して得られた報酬に基づいて，パラメータの更新方向を推定する．ES は報酬関数が微分不可能であっても適用可能であり，また並列化が容易であるという利点を持つ．

しかし，LLM に対して ES を適用する場合，計算コストが大きな障壁となる．従来の ES では，上述の通りノイズを加えたモデル全てにおいて，自己回帰的なトークン生成を行う必要がある．LLM の推論コストは非常に高く，学習時間の大部分がこの生成プロセスに費やされる．本論文は，ノイズを加えたモデルでの生成を行わず，現在のモデルで生成された良質なサンプルの「尤度」を用いてパラメータ更新方向を推定する Likelihood-Based Evolution Strategies（LBES）を提案し，その有効性を検証する．

2　関連研究

2.1　ES

ES は，勾配情報を直接必要としないブラックボックス最適化手法の一種である．Salimans ら [5] は，この手法を深層ニューラルネットワークの最適化に適用し，パラメータにノイズを加えた複数のモデルを用いて並列にタスクを実行することで，得られた報酬に基づいて勾配を推定するスケーラブルな枠組みを提案している．彼らは，各ワーカー間で巨大なパラメータベクトルではなく，ノイズ生成のシード値のみを通信することで，大規模な並列化が可能であることを示している．この特性により，ES はスパースな報酬や長い時間軸を持つタスクにおいて強化学習（RL）に匹敵，あるいは凌駕する性能を示している．

2.2　LLM に対する ES の適用

近年，LLM のアライメントやファインチューニングにおいても ES の適用が模索されている．Qiu ら [1] は，数十億パラメータ規模の LLM に対しても，全パラメータを対象とした ES によるファインチューニングが可能であることを示し，推論タスクにおいて RL 手法よりもサンプル効率や安定性に優れることを報告している．また，探索空間を絞った手法も提案されている．Jin ら [2] は LoRA [6] 行列の生成元となる低次元の潜在ベクトルを CMA-ES [7] で最適化している．Korotyshova ら [3] の ESSA は，LoRA と特異値分解を組み合わせ，最適化対象を特異値のみに絞ることで探索空間の次元を圧縮し，CMA-ES を適用している．Sarkar ら [4]

Algorithm 1 ES

Require: Batch size B, reward function $R(\cdot)$, total steps T, population size λ, noise scale σ, learning rate α, base parameters θ_0.

for $t = 1$ to T **do**
  Sample prompts $X \leftarrow \{x_1, \ldots, x_B\}$ from dataset
  for $k = 1$ to λ **in parallel do**
    Generate noise $\epsilon_k \sim \mathcal{N}(0, I)$
    Perturb parameters: $\theta^{(k)} \leftarrow \theta_{t-1} + \sigma\epsilon_k$
    Generate completions for each prompt: $Y^{(k)} \leftarrow \{y_i^{(k)}\}_{i=1}^{B} \sim \pi_{\theta^{(k)}}(\cdot|x_i)$
    Calculate rewards: $r_i^{(k)} \leftarrow R(y_i^{(k)})$ for all i
    Compute mean reward: $R_k \leftarrow \frac{1}{B}\sum_{i=1}^{B} r_i^{(k)}$
  end for
  Calculate advantages: $A_k \leftarrow \frac{R_k - \text{Mean}(R)}{\text{Std}(R)+\varepsilon}$ for $k = 1, \ldots, \lambda$ ▷ ε is a small constant
  Update parameters: $\theta_t \leftarrow \theta_{t-1} + \frac{\alpha}{\lambda\sigma}\sum_{k=1}^{\lambda} A_k\epsilon_k$
end for

の EGGROLL は，バッチ処理と低ランク行列を用いて大規模な個体数での学習を実現している．さらに，Huang ら [8] は，生成されたテキストの対数確率の勾配をノイズとして利用する ESO（Evolution Strategy Optimization）を提案している．

2.3 生成コストの課題と提案手法の位置付け

これらの先行研究 [1, 2, 3, 4] は，LLM に対する ES の有効性を示した点で重要であるが，RL タスクの学習においては共通して「ノイズを加えた各モデルを用いて実際にテキスト生成を行う」というプロセスを経ている．LLM の自己回帰的な生成は計算コストが高く，学習全体のボトルネックとなる．これに対し，本論文が提案する LBES は，ノイズ付加モデルでの生成を行わず，現在のモデルで生成・選抜された良質なサンプルに対する「尤度」のみを計算する点で決定的に異なる．これにより，生成に伴う膨大な計算コストを削減しつつ，ES による効率的な最適化を実現する．

3 手法

本論文は，LLM のブラックボックス最適化において，従来の ES における生成コストの課題を解消し，学習の高速化を実現する LBES を提案する．提案手法の全体像を図 1（右）に，詳細なアルゴリズムを Algorithm 2 に示す．

3.1 LBES

LBES は，計算コストの高い生成プロセスを現在のモデルのみに限定することで効率化を図る手法である．パラメータ空間における探索は，モデルによる生成結果の「良さ（報酬）」と，ノイズを与えたモデルにおけるその生成結果の「尤度」との相関に基づいて行う．具体的な手順は以下の通りである．

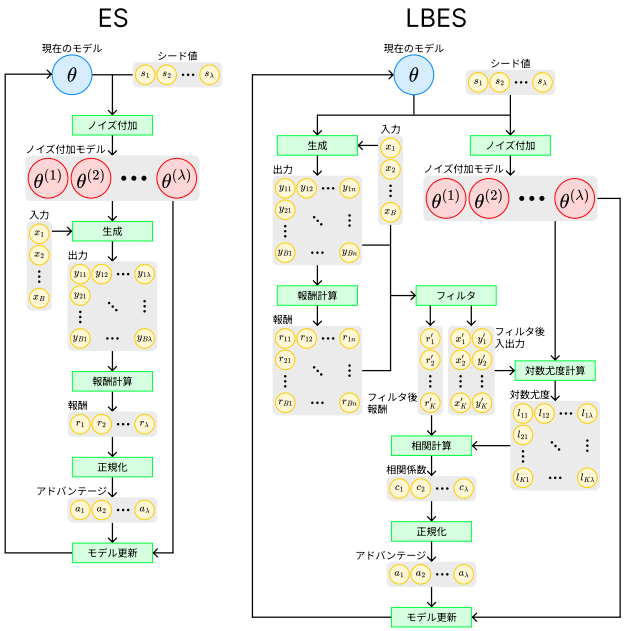

図 1 ES と LBES の処理フローの比較

まず，現在のパラメータ θ_{t-1} を持つモデルを用いて，各プロンプト x_i に対して n 個の回答 $\{y_{i,j}\}_{j=1}^{n}$ を生成する．ここで生成される回答は多様性を確保するため，温度付きサンプリング等を用いて取得する．次に，生成された各回答に対して報酬関数 $R(\cdot)$ を適用し，報酬 $r_{i,j}$ を計算する．さらに，得られた回答群から効果的な学習信号を抽出するため，報酬の分散に基づいたフィルタリングを行う．具体的には，まず各プロンプト x_i に対する n 個の回答の報酬の分散 $v_i = \text{Var}(\{r_{i,j}\}_{j=1}^{n})$ を計算する．報酬の分散が小さい（全ての回答が良い，または悪い）プロンプトはパラメータ更新への寄与が小さいため，分散が大きい上位 k_{var} 個のプロンプトのみを選抜する．

Algorithm 2 LBES

Require: Batch size B, samples per prompt n, reward function $R(\cdot)$, total steps T, population size λ, noise scale σ, learning rate α, base parameters θ_0, top-k prompts by reward variance k_{var}, top-k rewards per prompt k_{top}, bottom-k rewards per prompt k_{bottom}.

for $t = 1$ to T **do**
  Sample prompts $X \leftarrow \{x_1, \ldots, x_B\}$ from dataset
  Generate n completions for each prompt: $Y \leftarrow \{y_{i,j}\}_{i=1,j=1}^{B,n} \sim \pi_{\theta_{t-1}}(\cdot|x_i)$ ▷ Parallelized
  Calculate rewards: $r_{i,j} \leftarrow R(y_{i,j})$ for all i, j
  Compute reward variance per prompt: $v_i \leftarrow \mathrm{Var}\left(\{r_{i,j}\}_{j=1}^{n}\right)$
  Select prompts: $I \leftarrow \mathrm{TopK}\left(\{v_i\}_{i=1}^{B}, k_{\text{var}}\right)$
  Select completions per retained prompt: $J_i \leftarrow \mathrm{TopK}\left(\{r_{i,j}\}_{j=1}^{n}, k_{\text{top}}\right) \cup \mathrm{BottomK}\left(\{r_{i,j}\}_{j=1}^{n}, k_{\text{bottom}}\right)$ for $i \in I$
  Construct filtered flat set: $D_{\text{filtered}} \leftarrow \{(x_i, y_{i,j}) \mid i \in I,\ j \in J_i\}$
  Flatten rewards accordingly: $R_{\text{filtered}} \leftarrow \{r_{i,j} \mid i \in I,\ j \in J_i\}$
  for $k = 1$ to λ **in parallel do**
    Generate noise $\epsilon_k \sim \mathcal{N}(0, I)$
    Perturb parameters: $\theta^{(k)} \leftarrow \theta_{t-1} + \sigma\epsilon_k$
    Calculate mean log-likelihood: $L_k \leftarrow \left[\frac{1}{|y|}\sum_{l=1}^{|y|} \log \pi_{\theta^{(k)}}(y_l|x, y_{<l})\right]_{(x,y)\in D_{\text{filtered}}}$
  end for
  Compute correlation coefficient: $C_k \leftarrow \mathrm{Corr}(R_{\text{filtered}}, L_k)$ for $k = 1, \ldots, \lambda$ ▷ Pearson correlation coefficient
  Calculate advantages: $A_k \leftarrow \frac{C_k - \mathrm{Mean}(C)}{\mathrm{Std}(C)+\varepsilon}$ for $k = 1, \ldots, \lambda$ ▷ ε is a small constant
  Update parameters: $\theta_t \leftarrow \theta_{t-1} + \frac{\alpha}{\lambda\sigma}\sum_{k=1}^{\lambda} A_k\epsilon_k$
end for

さらに，選抜された各プロンプトについて，報酬が高い上位 k_{top} 個と下位 k_{bottom} 個のサンプルを抽出し，これらを D_{filtered} として構築する．これにより，モデルの改善余地が大きく，かつ良否の対比が明確なデータのみを学習に利用する．

続いて，パラメータの更新方向を決定するために λ 個のノイズベクトル ϵ_k を生成し，ノイズのスケール σ を用いてノイズが付加されたパラメータ $\theta^{(k)} = \theta_{t-1} + \sigma\epsilon_k$ を用意する．ここで，従来の ES とは異なり，各ノイズ付加モデル $\theta^{(k)}$ で新たな回答生成は行わない．代わりに，現在のモデルで生成・選抜された回答データ D_{filtered} に対する対数尤度を計算する．具体的には，ある回答 y が，ノイズを加えたモデル $\theta^{(k)}$ においてどれだけの尤度で生成されうるかを計算する．もし，あるノイズ方向 ϵ_k が「高報酬な回答の尤度を高め」かつ「低報酬な回答の尤度を下げる」傾向にあれば，その方向はパラメータ更新において好ましい方向であると判断できる．

最終的に，選抜されたサンプルの報酬値 R_{filtered} と，各ノイズ付加モデルにおける対数尤度 L_k との間のピアソン相関係数 C_k を算出する．この相関係数を正規化したものを A_k とし，学習率 α を用いて以下の式に従ってパラメータを更新する．

$$\theta_t \leftarrow \theta_{t-1} + \frac{\alpha}{\lambda\sigma}\sum_{k=1}^{\lambda} A_k\epsilon_k \tag{1}$$

3.2 ES と LBES の比較

図 1 に，従来の ES（左）と提案手法である LBES（右）の処理フローの比較を示す．

従来の ES では，ノイズを付加した λ 個のモデルのそれぞれを用いて，実際に回答の生成を行う必要がある．LLM においてトークンを自己回帰的に生成する処理は計算コストが高い．

一方，LBES では，回答の生成を行うのは現在のモデル（図中の青色のモデル）のみである．ノイズを付加したモデル群は，すでに生成されたトークン列に対する対数尤度を計算するだけでよい．この対数尤度の計算は，LLM の順伝播を一度通すだけで計算可能であり，自己回帰的な生成と比較して高速である．また，生成された回答の中から報酬に基づいてフィルタリングを行うことで，尤度計算の対象を絞り込む効率化も図っている．

このように，LBES は「生成」と「評価」のプロセスを分離し，高コストな生成プロセスを最小限に抑えている点が最大の特徴である．

4 実験

提案手法 LBES の有効性を検証するため，GSM8K[1)] およびカウントダウンタスクを用いた比較実験を行う．

1) https://huggingface.co/datasets/openai/gsm8k

4.1 設定

GSM8K：システムプロンプトは付録の表 3 に示す．報酬関数は，フォーマット報酬（0.1 点）と，抽出された数値の正答報酬（1 点）の和とする．

カウントダウンタスク：与えられた複数の整数から四則演算によりターゲット数値を導出するタスクである．本実験では，入力する整数を 3 つ（範囲：1〜99），ターゲットを 1〜999 の範囲とし，データセットとして訓練用 8,192 件，テスト用 1,024 件を生成する．プロンプトを付録の表 4 に示す．報酬関数は，フォーマット報酬（0.1 点）と，抽出された式の正答報酬（1 点）の和とする．

モデルと環境：実験には `meta-llama/Llama-3.1-8B-Instruct`[2] をベースモデルとして使用する．推論ライブラリは vLLM[9] を用いる．計算環境には NVIDIA H100 SXM5（94GB HBM2e）を 4 基搭載したノードを用いる．

ハイパーパラメータ：バッチサイズ $B = 64$，個体数 $\lambda = 32$（4 並列），学習率 $\alpha = 5 \times 10^{-7}$，ノイズ標準偏差 $\sigma = 1 \times 10^{-3}$ は共通設定とする．タスクごとの設定として，エポック数は GSM8K で 1，カウントダウンタスクで 2 とし，最大生成トークン数はそれぞれ 1,024，2,048 とする．手法ごとの設定は両タスクで共通であり，ES では $n = 1$（温度 0），LBES では $n = 8$（温度 0.1）で生成を行う．また LBES のフィルタリングは，分散の上位 $k_{\text{var}} = 32$ 件を選抜し，各プロンプトから報酬の上位・下位各 1 件（$k_{\text{top}} = 1, k_{\text{bottom}} = 1$）を利用する．

4.2 結果

評価指標にはテストセットにおける精度を用いた．なお，学習前のベースモデルの精度は GSM8K で 66.0%，カウントダウンタスクで 11.9%であった．

両タスクにおける訓練結果を表 1，2 に示す（学習曲線は付録の図 2 および図 3 を参照）．実験の結果，LBES は一定の精度に到達するまでの時間を ES と比較して大幅に短縮できることが確認された．一方で，十分な学習時間を確保した場合の最高精度に関しては，両タスクともに ES が LBES を上回る結果となった．特にカウントダウンタスクにおいては，ES が最高精度 82.8%に達したのに対し，LBES は 55.1%にとどまった．

2） https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

表 1 GSM8K の訓練結果

	精度 80%到達訓練時間（秒）	最高精度（%）
ES	669	87.3
LBES	222	85.1

表 2 カウントダウンタスクの訓練結果

	精度 50%到達訓練時間（秒）	最高精度（%）
ES	13382	82.8
LBES	5685	55.1

4.3 考察

学習速度について：LBES による学習時間の短縮は，各ノイズ付加モデル（$\lambda = 32$）において完全なテキスト生成を行う代わりに，現在のモデルで生成されたサンプルに対する尤度計算のみを行った効果である．LLM においては自己回帰的な生成処理が計算のボトルネックとなるため，このコストを削減できたことの寄与は大きい．

精度について：LBES の最高精度が ES と比較して低い原因として，LBES ではノイズ付加モデル自体が生成したサンプルを評価していない点が考えられる．すなわち，ノイズ付加モデルにおいて「正解データの尤度が高い」ことと「正解データを正しく生成できる」ことの間には乖離が存在する場合がある．また，現在のモデルの生成分布に依存したサンプルのみで学習を行うため，探索範囲が限定的になる可能性も示唆される．今後は，定期的に生成を行うハイブリッドな手法などの検討が課題である．

5 おわりに

本論文は，LLM の進化戦略による最適化において，生成コストの問題を解決するための LBES を提案した．実験により，LBES は従来の ES と比較して，一定精度への到達時間を大幅に短縮できることを示した．その一方で，最終的な到達精度においては ES に及ばない場合があるという課題も明らかになった．今後は，この速度と精度のトレードオフを解消するためのアルゴリズムの改良や，より大規模なモデルおよび多様なタスクでの検証を進める予定である．

謝辞

本研究は JST CREST JPMJCR2565 の支援を受けた．また，東京科学大学のスーパーコンピュータ TSUBAME4.0 を利用して実施した．

参考文献

[1] Xin Qiu, Yulu Gan, Conor F. Hayes, Qiyao Liang, Elliot Meyerson, Babak Hodjat, and Risto Miikkulainen. Evolution strategies at scale: Llm fine-tuning beyond reinforcement learning. arXiv, 2025. abs/2509.24372.

[2] Feihu Jin, Yifan Liu, and Ying Tan. Derivative-free optimization for low-rank adaptation in large language models. **IEEE/ACM Trans. Audio, Speech and Lang. Proc.**, Vol. 32, p. 4607â4616, October 2024.

[3] Daria Korotyshova, Boris Shaposhnikov, Alexey Malakhov, Alexey Khokhulin, Nikita Surnachev, Kirill Ovcharenko, George Bredis, Alexey Gorbatovski, Viacheslav Sinii, and Daniil Gavrilov. Essa: Evolutionary strategies for scalable alignment. arXiv, 2025. abs/2507.04453.

[4] Bidipta Sarkar, Mattie Fellows, Juan Agustin Duque, Alistair Letcher, Antonio LeÃ³n Villares, Anya Sims, Dylan Cope, Jarek Liesen, Lukas Seier, Theo Wolf, Uljad Berdica, Alexander David Goldie, Aaron Courville, Karin Sevegnani, Shimon Whiteson, and Jakob Nicolaus Foerster. Evolution strategies at the hyperscale. arXiv, 2025. abs/2511.16652.

[5] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. arXiv, 2017. abs/1703.03864.

[6] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In **International Conference on Learning Representations**, 2022.

[7] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. **Evolutionary Computation**, Vol. 9, No. 2, pp. 159–195, 2001.

[8] Bo Huang, Yuxin Jiang, Mingyang Chen, Yi Wang, Hongyang Chen, and Wei Wang. When evolution strategy meets language models tuning. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, **Proceedings of the 31st International Conference on Computational Linguistics**, pp. 5333–5344, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics.

[9] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In **Proceedings of the 29th Symposium on Operating Systems Principles**, SOSP '23, p. 611â626, New York, NY, USA, 2023. Association for Computing Machinery.

A 付録

表 3 GSM8K のシステムプロンプト

You are a reasoning model designed to think step-by-step before providing answers. Always follow this format:

1. Start your response with your thinking process inside ‘<think></think>‘ tags
2. After your thinking, provide your explanation and include the final numerical answer inside ‘<answer></answer>‘ tags

In your ‘<think>‘ section:
- Break down the problem into smaller parts
- Consider different approaches or perspectives
- Work through your reasoning step by step
- Identify any assumptions you’re making
- Check your logic for potential errors

Your thinking should be thorough but concise. After closing the ‘</think>‘ tag, provide your explanation and wrap the numerical answer within ‘<answer></answer>‘ tags. The answer tags should contain only the number.

表 4 カウントダウンタスクのプロンプト

システムプロンプト	# Identity You are a Mathematical Reasoning Expert specialized in solving arithmetic puzzles. Your task is to use ALL the given numbers EXACTLY ONCE with basic arithmetic operations to reach the target number. # Instructions - You MUST use each given number EXACTLY ONCE, and NO other numbers - Use ONLY the operations: +, -, *, / - Begin with your reasoning process inside <think></think> tags - After the reasoning, provide your final answer inside <answer></answer> tags (e.g., <answer>(1 + 2) / 3</answer>) - The answer content must use ONLY digits, operators, parentheses, and spaces, EXCLUDING ”=” and the result
ユーザープロンプト例	Given numbers: 13, 8, 50 Target number: 54

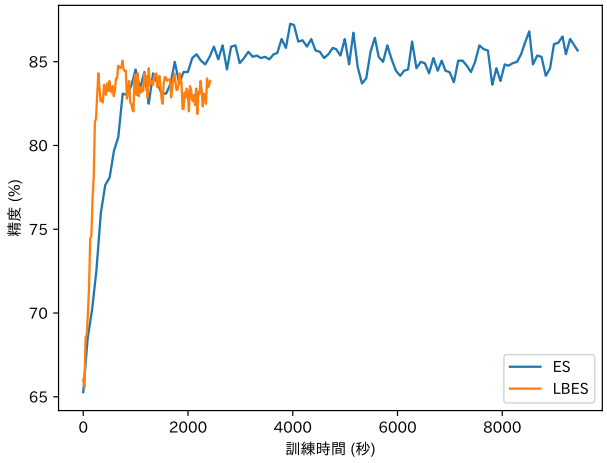

図 2 GSM8K の訓練時間と精度

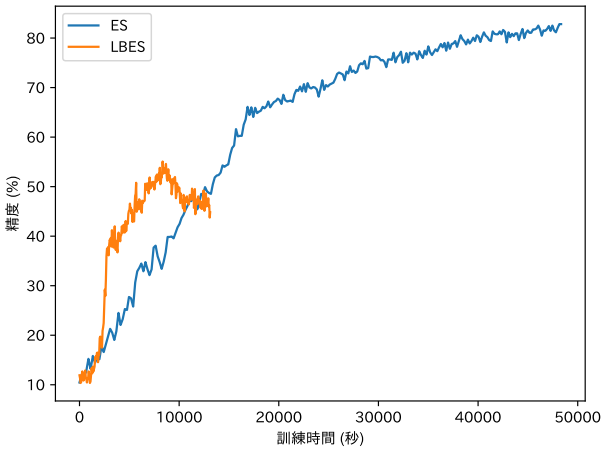

図 3 カウントダウンタスクの訓練時間と精度